IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application No.: 10/821,431	§	Examiner:	Vincent Lai
Filed: April 9, 2004	§	Group/Art Unit:	2181
Inventor(s):	§	Atty. Dkt. No:	5181-96100
Cypher, et al.	§		
Title: Branch Prediction Scheme Utilizing Multiple Branch Predictors and Multiple Index Hashing Mechanisms for Determining a Final Prediction Based on Intermediaries	§		

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir/Madam:

Further to the Notice of Appeal filed March 9, 2007, Appellant presents this Appeal Brief. Appellant respectfully requests that this appeal be considered by the Board of Patent Appeals and Interferences.

I. REAL PARTY IN INTEREST

The present application is owned by Sun Microsystems, Inc. An assignment of the present application to the owner is recorded at Reel 015214, Frame 0253.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences known to Appellant.

III. STATUS OF CLAIMS

Claims 1-4, 6-11, 13-17, 19-24, and 26-27 are pending. Claims 5, 12, 18, and 25 are cancelled. Claims 1-4, 6-11, 13-17, 19-24, and 26-27 are rejected under 35 U.S.C. § 103(a). It is these rejections that are being appealed. A copy of claims 1-4, 6-11, 13-17, 19-24, and 26-27 is included in the Claims Appendix attached hereto.

IV. STATUS OF AMENDMENTS

No unentered amendment to the claims has been filed after final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Many high performance pipelined microprocessors maintain their performance by keeping the various pipelines as full as possible. However, certain control transfer instructions may cause various stages within a pipeline to be flushed and the instructions in those stages to be canceled or discarded. One such instruction is a branch instruction. A branch instruction is an instruction which causes subsequent instructions to be fetched from a target address identifying an instruction stream beginning at an arbitrary location in memory. Unconditional branch instructions always branch to the target address, while conditional branch instructions may select either the next sequential address or the target address based on a condition such as the outcome of a prior instruction. Thus, when instructions are prefetched into a processor's instruction-processing pipeline sequentially, and a conditional branch is taken, the contents of early stages of the pipeline may contain instructions that should not be executed but rather should be flushed.

Accordingly, in conjunction with the pre-fetching of instructions, it is generally beneficial to predict whether a conditional branch instruction, when executed, will be taken or not. For this purpose, many pipelined microprocessors employ a branch prediction mechanism.

Many branch prediction mechanisms have been implemented that yield relatively good results. One such branch prediction mechanism is the GShare branch prediction mechanism. The GShare mechanism combines a global branch history and several bits of the program counter (PC) of a current branch instruction (e.g., fetch address of the current branch instruction) to form an index into a predictor table. Each entry in the predictor table typically stores a counter value that may be indicative of whether a branch should be taken or not. Although the GShare mechanism may be an effective prediction mechanism, aliasing of index values is a concern. For example, the GShare mechanism may produce the same index value and therefore an erroneous prediction, when provided with different input combinations of the PC and global branch history that correspond to different branches.

Independent claim 1 is directed to a branch prediction mechanism that may include a first storage (*See* for example, FIG. 2, 260; FIG. 3, 360) including a first plurality of locations for storing a first set of partial prediction information. The branch prediction mechanism may also include a second storage (*See* for example, FIG. 2, 270; FIG. 3, 370) including a second plurality of locations for storing a second set of partial prediction information (*See* page 7, lines 22-27; page 9, lines 15-22). Further the branch prediction mechanism may include a control unit (*See* for example, FIG. 2, 290; FIG. 3, 390) that performs a first hash function (*See* for example, FIG. 2, 230; FIG. 3, 330) on input branch information to generate a first index for accessing a selected location within the first storage (*See* page 8, line 25 - page 9, line 5). In one particular implementation, the input branch information may include branch history information in addition to branch address information such as the fetch address of current branch instruction, for example (*See* page 8, lines 1-17). The control unit also performs a second hash function (*See* for example, FIG. 2, 240; FIG. 3, 340) on the input branch information to generate a second index for accessing a selected location within the second storage (*See* page 8, line 25 - page 9, line 5). The control unit may further provide a prediction value based on corresponding partial prediction information in the selected locations of the first and the second storages (*See* page 9, lines 15-25). Further, the control unit may update the selected locations of the first and the second storages dependent on whether the prediction value yields an accurate branch prediction (*See* page 10, lines 8-17).

Independent claim 14 is directed to a method of predicting branches. The method includes storing a first set of partial prediction information within a first storage including a first plurality of locations and storing a second set of partial prediction information within a second storage including a second plurality of locations (*See* page 7, lines 22-27; page 9, lines 15-22). The method also includes performing a first hash function on input branch information to generate a first index for accessing a selected location within the first storage, and performing a second hash function on the input branch information to generate a second index for accessing a selected location within the second storage (*See* page 8, line 25 - page 9, line 5). The input branch information may include address information corresponding to a fetch address of a current branch instruction (*See* page 8, lines 1-17). The method also includes providing a prediction value based on corresponding partial prediction information in the selected locations of the first and the second storages (*See* page 9, lines 15-25). The method further includes updating the selected locations of the first and the second storages dependent on whether the prediction value yields an accurate branch prediction (*See* page 10, lines 8-17).

Independent claim 27 is a means plus function claim directed toward a branch prediction mechanism including means for storing a first set of partial prediction information within a first storage including a first plurality of locations, and means for storing a second set of partial prediction information within a second storage including a second plurality of locations (For example, *See* page 7, lines 22-27 and page 9, lines 15-

22, where prediction control unit 290 may use storages 260 and 270 to provide this functionality). The mechanism also includes means for performing a first hash function on input branch information to generate a first index for accessing a selected location within the first storage, and performing a second hash function on said input branch information to generate a second index for accessing a selected location within the second storage (For example, *See* page 8, line 25 - page 9, line 5, where prediction control unit 290 may use hash functions 230 and 240 to provide this functionality). The input branch information may include address information corresponding to a fetch address of a current branch instruction. In addition, the mechanism includes means for providing a prediction value based on corresponding partial prediction information in the selected locations of the first and the second storages (For example, *See* page 9, lines 15-25, where prediction control unit 290, may provide this functionality). The mechanism further includes means for updating the selected locations of the first and the second storages dependent on whether the prediction value yields an accurate branch prediction (For example, *See* page 10, lines 8-17, where prediction control unit 290, may provide this functionality).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 1-4, 6-8, 13-17, 19-20, and 26-27 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Loh (U.S. Patent Publication No. 2005/0223203) (hereinafter "Loh") in view of McFarling (U.S. Patent Publication No. 2001/0056531) (hereinafter "McFarling").

2. Claims 9-11 and 22-24 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Loh in view of McFarling, and in further view of Yeh (U.S. Patent No. 6,427,206) (hereinafter "Yeh").

VII. ARGUMENT

First Ground of Rejection:

Claims 1-4, 6-8, 13-17, 19-20, and 26-27 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Loh in view of McFarling. Appellant traverses this rejection for at least the following reasons.

Independent claims (by number):

Appellant respectfully submits that each of claims 1, 14, and 27 recites a combination of features not taught or suggested in Loh and McFarling. For example, claim 1 recites a combination of features including: "a first storage including a first plurality of locations for storing a first set of partial prediction information,"a second storage including a second plurality of locations for storing a second set of partial prediction information," "...perform a first hash function on input branch information to generate a first index for accessing a selected location within said first storage and to perform a second hash function on said input branch information to generate a second index for accessing a selected location within said second storage, wherein said input branch information includes address information corresponding to a fetch address of a current branch instruction," "... to provide a prediction value based on corresponding partial prediction information in said selected locations of said first and said second storages," and "… to update said selected locations of said first and said second storages dependent on whether said prediction value yields an accurate branch prediction," as recited in Appellant's claim 1.

Appellant respectfully disagrees with the Examiner's characterization of both Loh and McFarling, and the allegation that the combination teaches the invention recited in Appellant's claim 1.

More particularly, in the final Office action dated, November 9, 2006, the Examiner alleges that Loh teaches a first storage including a first plurality of locations for storing a first set of partial prediction information (Loh Fig. 4, element 405 (e.g., branch predictor 1) and a second storage including a second plurality of locations for storing a second set of partial prediction information (Loh Fig. 4, element 405 (e.g., branch predictor 2). Appellant respectfully disagrees. Specifically, Loh teaches at paragraphs [0018], [0019], and [0022]

> "…The prediction history information may be accessed in segments by a number of intermediate branch prediction units 405.
>
> [0019] In one embodiment of the invention, four intermediate branch history units access four segments of branch history from the branch history register. However, in other embodiments, the number of segments and corresponding intermediate branch history units may be greater or fewer than four. In some embodiments of the invention, some intermediate branch history units may be in parallel and others may be in series with any of the parallel branch history units. Furthermore, the series intermediate branch history units may perform intermediate branch predictions in parallel with each other in other embodiments of the invention.
>
> [0022] FIG. 5 is a flow diagram illustrating a method for performing at least one embodiment of the invention. In operation 501, a number of branch prediction segments are accessed in parallel. At operation 505, a number of intermediate branch predictions are performed based off of the branch prediction segments, in which each intermediate branch prediction is based off of a different branch history segment and each branch history segment is smaller than the sum of the branch history segments. At operation 510, a final branch prediction is made based off of the intermediate branch predictions." (Emphasis added)

From the foregoing, Appellant asserts Loh merely teaches the branch predictor units 405 making predictions using information obtained from the segment registers 401, and the final predictor making a prediction based off the intermediate predictions. Appellant submits Loh does not teach disclose or suggest that the branch predictor units 405 have any storage capacity whatsoever. Moreover Appellant cannot find any reference in Loh, whether implied or otherwise, that each predictor 405 may be a storage including a plurality of locations for storing a set of predictions. Appellant again notes that Loh does not provide any specific detail with regard to the structure of elements 405, other than what is discussed above.

In the Advisory action dated March 2, 2007 the Examiner asserts "the Examiner contends the branch predictor units 405 must have some kind of storage capacity. In paragraph 18, Loh teaches that branch history is accessed, meaning there must be some sort of storage element available for the branch predictor." Appellant submits that just because the Examiner contends something is true does not mean that it is without some reasonable teaching in the art or evidentiary support for the allegation. Appellant disagrees with the Examiner and submits Loh actually discloses at paragraph [0018]

> "a prediction history register 401, in which prediction history is stored in one embodiment of the invention. The prediction history register may also be a memory location instead of a register within the processor or some combination thereof." (Emphasis added)

Thus, the branch history information is stored in the prediction history registers 401 and not the branch predictor 405. In addition, as illustrated in FIG. 4, Loh discloses one segment register 401 per branch predictor unit 405.

Thus, Appellant submits Loh **does not teach or suggest** "a first storage including a first plurality of locations for storing a first set of partial prediction information," and "a second storage including a second plurality of locations for storing a second set of partial prediction information," as recited in Appellant's claim 1.

In the final Office action dated, November 9, 2006, the Examiner acknowledges that Loh does not disclose the hashing functions as recited in Appellant's claim 1. However, the Examiner asserts McFarling teaches the limitation. Appellant respectfully disagrees. More particularly, the Examiner asserts McFarling teaches, at paragraph [0026], lines 10-13; and fig. 14, paragraphs [0087]-[0089], a control unit configured to perform a first hash function on input branch information to generate a first index for accessing a selected location within said first storage, and to perform a second hash function on said input branch information to generate a second index for accessing a selected location within said second storage.

In addition, the Examiner asserts the case of performing multiple has functions in which multiple storages can be accessed is shown. Appellant notes that while McFarling teaches branch predication, and the use of hash functions, Appellant respectfully disagrees with the Examiner's assertion that McFarling teaches the limitations as recited in Appellant's claims. It appears the Examiner is asserting that McFarling is describing the use of a single hash function (e.g., fig. 5, para [0026]), and the multiple hash functions (2) shown in fig. 14 are an extension of that. Appellant disagrees, and notes that McFarling does teach using a hash function in FIG. 5. However, Appellant asserts this is the GShare branch prediction mechanism described in Appellant's background section. Further, Appellant asserts the use of the multiple hash functions as shown in Fig. 14 of McFarling, is not a direct extension of the use of one hash function.

Specifically, McFarling teaches

> "The present invention provides <u>a serial branch predictor that includes a first component predictor operating according to a first algorithm to predict an action, and any number of subsequent component predictors operating according to alternate algorithms to predict the action</u>. The predictor further includes means, coupled to each predictor, for choosing between the subsequent predictors to provide a refined prediction of the action from the serial branch predictor. Such an arrangement provides a better prediction mechanism, since it serially combines multiple component predictors with varying characteristics to overrule the prediction from any prior component predictor if and only if an improvement in prediction accuracy is likely. <u>Each subsequent stage therefore focuses on correction of predictions made by a prior stage.</u> <u>In the</u>

preferred embodiment, known as the SerialBLG predictor, the first predictor algorithm is bimodal, a second predictor algorithm is local, and a third predictor algorithm is global. Further, each stage is improved according to various methods." (*See* Summary) (Emphasis added)

McFarling also illustrates in FIG. 14, and teaches at paragraphs [0085]-[0091]

"Global Stage Indexing

[0086] FIG. 14 shows the global stage predictor used in the best mode SerialBLG predictor, which makes use of the stew code, conditional prediction, and partial dominance. The global information is stored in a stew register 140 as described above. The stew register value is XOR'ed with the address of the branch on line 141 to produce V on line 142: V=Stew XOR Addr.

[0087] The previous stage prediction on line 143 is appended to this value to implement conditional prediction: V.sup.+=V, pred.sub.-1

[0088] The value V.sup.+ provides the basis for making the global stage prediction. V.sup.30 could be used to directly access an array of counters, however, as suggested above, most of these counters would be unnecessary. A better approach is to simulate a large array of counters using a cache mechanism 144. On a cache miss, it is assumed that the unavailable count would agree with the prior stage prediction. Unavailable counters are added to the cache only if the final prediction value on line 145 is wrong.

[0089] On such a replacement, the appropriate tag is set to correspond to the V.sup.30 value, and the counter 146 is initialized to weakly agree with the branch causing the miss. If the cache in the global predictor stage uses LRU replacement, the LRU order is only affected when a counter is useful, i.e. generates a better prediction than the earlier stage.

[0090] Cache Tag Hashing

[0091] The method of indexing the global predictor stage cache is particularly significant. For good performance, set-associative caches require addresses that spread out fairly uniformly across the cache. For example, with a 2-way set-associative cache, if everything maps to the same set, then no more than two things can be stored, no matter how large the cache is. The V.sup.+ value, even using the stew code, is still less uniformly distributed than is preferred. To improve performance, two steps can be taken. First, the high order bits on line 147 of V.sup.+ are more random than the low order bits on line 148 because the high order bits on line 147 are a function of a long sequence of branches, whereas the

> low order bits on line 148 are a function only of the last few branches. It is therefore better to use the low order bits on line 148 for the tag value T on line 149. Second, to further increase the randomness of the remaining bits, the tag value T on line 149 can be XOR'ed into these remaining bits, Z on line 147, to obtain the set index I on line 150:…"

From the foregoing, it appears McFarling is using two hash functions (XOR) to access a single cache structure 144. In addition, Appellant notes the second hash function is operating on at least a portion of the result from the first hash function. This is in contrast to the assertions made by the Examiner, and to Appellant's recited claim language.

In the Advisory action dated March 2, 2007, the Examiner noted "McFarling is used to teach the obviousness of using hashing functions and that it is unreasonable to believe that McFarling and Loh can be combined without adaptations. Loh teaches multiple cache structures that can utilize hash functions and thus to properly combine Loh and McFarling, one having ordinary skill in the art would utilize more than one hash function."

Appellant further disagrees with the Examiner's above assertion and similar assertions made ion previous Office actions. Appellant does not doubt the utility of using a hashing function. However, as disclosed in Appellant's background, there are index aliasing problems associated with the GShare mechanism of McFarling, which Appellant addresses. In addition, as discussed above, McFarling uses the hashing functions differently than Appellant, regardless of whether the Examiner concedes the point, McFarling does, in fact, use the two hash functions to access one cache structure, by hashing different information for a different reason. Thus, Appellant submits the Examiner has not established a *prima facie* case of obviousness simply by showing that McFarling uses hash function(s). Appellant further submits that due to the above-described differences and using hindsight obtained by Appellant's disclosure, the combination of Loh and McFarling could not be used without undue experimentation.

Appellant submits neither Loh nor McFarling, taken either singly or in combination, teach or suggest the combination of features recited in Appellant's claim 1. Appellant's claims 14 and 27 recite features that are similar to the features recited in Appellant's claim 1.

For at least the above stated reasons, Appellant submits that the rejection of claims 1, 14, and 27 is in error and requests reversal of the rejection. The rejection of claims 3-5 (dependent from claim 1), claims 18-20 (dependent from claim 12), claims 23-25 (dependent from claim 22), and claim 35 (dependent from claim 29) are similarly in error for at least the above stated reasons, and reversal of the rejection is requested. Each of claims 3-5, 18-20, 23-25, and 35 recite additional combinations of features not taught or suggested in the cited art.

<u>**Separately argued dependent claims (by number)**</u>

Claims 2 and 8, and 15 and 21 depend from claims 1 and 14, respectively. Accordingly, the rejection of claims 2, and 8, and 15 and 21 is in error for at least the reasons highlighted above with regard to claims 1 and 14.

In addition, Appellant respectfully disagrees with many of the Examiner's assertions with regard to dependent claims 2-13 and 15-26. For example, in several of the dependent claim rejections, the Examiner indicates Loh discloses "the first hash function" and "the second hash function …" Appellant submits the Examiner has already conceded that Loh does not teach any hash functions.

Specifically, in regard to claim 2, the Examiner asserts Loh teaches "wherein said prediction value provides a strongly/weakly taken/not taken branch prediction indication that is indicative of whether the current branch instruction is taken upon execution." The Examiner asserts Loh teaches this at paragraph [0020]. Appellant asserts Loh ONLY teaches "a final branch history predictor unit 410 to generate <u>a final branch prediction as function of the intermediate branch predictions performed by the intermediate branch prediction units</u>." Loh is silent as to how this is performed. The Examiner is merely speculating that Loh teaches how the predictions are performed.

With regard to claim 8, the Examiner asserts that Loh discloses "using multiple values of saturating counters (typically ranging between the value of 0-3 as common at the time) and performing an action on said counters to derive a final prediction. Appellant can find absolutely <u>no teaching of this feature in Loh</u>. Appellant accordingly disagrees that it would not be obvious to generate "said prediction value…by summing respective counter values stored within said selected location within said first storage and said selected location within said second storage" as recited in claims 8 and 11.

Appellant's claims 15 and 21 recite features that are similar to the feature recited in claims 2 and 8, respectively. Accordingly, similar arguments are maintained for claims 15 and 21.

For at least the above stated reasons, Appellant submits that the rejections of claims 2, 8, 15 and 21 are in error and requests reversal of the rejection.

<u>Second Ground of Rejection</u>:

Claims 9-11 and 22-24 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Loh in view of McFarling, and in further view of Yeh. Claims 9-11 and 22-24 depend from claims 1 and 14, respectively. Accordingly, the rejection of claims 9-11 and 22-24 is in error for at least the reasons highlighted above with regard to claims 1 and 14.

<u>Separately argued dependent claims (by number)</u>

Appellant submits claims 10, 11, 23, and 24 recite fetaures that are not taught or suggested by the cited references.

With regard to claim 10, the Examiner asserts that Yeh discloses at col. 6, lines 18-25 "each of said first and said second sets of partial prediction information includes a plurality of counter values each corresponding to a strongly/weakly agree/disagree indication that is indicative of whether said branch prediction hint bit embedded within said current branch instruction is to be used by said control unit. The Examiner note states "Yeh discloses using a hardware branch predictor if confidence in the compiler hint is not strong…"

Appellant can find absolutely <u>no teaching of this feature in Yeh</u>. Appellant submits Yeh teaches at col. 6, lines 18-25

> "When a "static not-taken" branch is encountered, typically there is no instruction flow change. Therefore, there is no need to cache that branch in the BPT. <u>When the information encoded in an instruction indicates that there is a likely taken or likely not taken prediction, the compiler is not very sure whether the branch should be taken or not take</u>n. In this case, <u>BPT 140 is used for dynamically predicting the behavior of branche</u>s. Dynamic prediction utilizes the history of a branch, recorded in the BPT, to predict the outcomes of upcoming branches. If the microprocessor that implements the embodiment of the present invention described herein has BPT prediction--i.e. dynamic prediction available, it may utilize the BPT prediction to determine whether a branch should be taken or not taken. If the BPT prediction is not available or the microprocessor wants to override the BPT prediction, the microprocessor may use the taken/not-

> taken field of the two-bit compiler hint to determine whether a branch should be taken or not taken. In one embodiment of the present invention, the two encoded bits are part of a 41 bit instruction, and are written by the compiler into the instruction, but the present invention is not limited in scope to 41-bit instructions. The length of an instruction is such that in each instruction there are at least two bits dedicated to encode the compiler hints."

Appellant submits Yeh is teaching using the HW branch predictor if the hint confidence is not strong enough. This is not the same as "a plurality of counter values each corresponding to a strongly/weakly agree/disagree <u>indication that is indicative of whether said branch prediction hint bit embedded within said current branch instruction is to be used by said control unit</u>."

Further, in regard to claim 11, Appellant's claim 11 recites "wherein said prediction value is generated by summing respective counter values stored within said selected location within said first storage and said selected location within said second storage."

The Examiner asserts none of the cited references teach this feature. However, the Examiner asserts since McFarling discloses "performing an action" on the counters to derive a final prediction, it would be obvious to do so.

Appellant respectfully disagrees that it would be obvious to generate "said prediction value…<u>by summing respective counter values</u> stored within said selected location within said first storage and said selected location within said second storage" as recited in claims 8 and 11.

For at least the above stated reasons, Appellant submits that the rejections of claims 10 and 11 are in error and requests reversal of the rejection.

Appellant's claims 23 and 24 recite features that are similar to the features recited in claims 10 and 11, respectively. Accordingly, similar arguments are maintained for claims 23 and 24.

CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejections of claims 1-4, 6-11, 13-17, 19-24, and 26-27 are erroneous, and reversal of the decision is respectfully requested.

The Commissioner is authorized to charge any fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-96100/SJC.

Respectfully submitted,

/Stephen J. Curran/
Stephen J. Curran
Reg. No. 50,664
AGENT FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
(512) 853-8800

Date: May 10, 2007

VIII. CLAIMS APPENDIX

The claims on appeal are as follows.

1. A branch prediction mechanism comprising:

a first storage including a first plurality of locations for storing a first set of partial prediction information;

a second storage including a second plurality of locations for storing a second set of partial prediction information; and

a control unit configured to perform a first hash function on input branch information to generate a first index for accessing a selected location within said first storage and to perform a second hash function on said input branch information to generate a second index for accessing a selected location within said second storage, wherein said input branch information includes address information corresponding to a fetch address of a current branch instruction;

wherein said control unit is further configured to provide a prediction value based on corresponding partial prediction information in said selected locations of said first and said second storages; and

wherein said control unit is further configured to update said selected locations of said first and said second storages dependent on whether said prediction value yields an accurate branch prediction.

2. The branch prediction mechanism as recited in claim 1, wherein said prediction value provides a strongly/weakly taken/not taken branch prediction indication that is indicative of whether the current branch instruction is taken upon execution.

3. The branch prediction mechanism as recited in claim 1, wherein said input branch information includes branch history information corresponding to an outcome of a number of preceding branch instructions.

4. The branch prediction mechanism as recited in claim 3, wherein each of said first hash function and said second hash function is configured to operate on a portion of said branch history information.

6. The branch prediction mechanism as recited in claim 1, wherein each of said first hash function and said second hash function is configured to operate on a different subset of bits of said fetch address.

7. The branch prediction mechanism as recited in claim 1, wherein each of said first and said second sets of partial prediction information includes a plurality of counter values each corresponding to a strongly/weakly taken/not taken branch prediction indication that is indicative of whether the current branch instruction is taken upon execution.

8. The branch prediction mechanism as recited in claim 7, wherein said control unit is further configured to use said prediction value to determine whether the current branch instruction is taken upon execution, wherein said prediction value is generated by summing respective counter values stored within said selected location within said first storage and said selected location within said second storage.

9. The branch prediction mechanism as recited in claim 1, wherein said control unit is further configured to use said prediction value to control whether a branch prediction is performed in accordance with a branch prediction hint encoded within the current branch instruction.

10. The branch prediction mechanism as recited in claim 9, wherein each of said first and said second sets of partial prediction information includes a plurality of counter

values each corresponding to a strongly/weakly agree/disagree indication that is indicative of whether said branch prediction hint bit embedded within said current branch instruction is to be used by said control unit.

11. The branch prediction mechanism as recited in claim 10, wherein said prediction value is generated by summing respective counter values stored within said selected location within said first storage and said selected location within said second storage.

13. The branch prediction mechanism as recited in claim 1 further comprising a third storage including a third plurality of locations for storing a third set of partial prediction information and wherein said control unit is further configured to perform a third hash function on said input branch information to generate a third index for accessing a selected location within said third storage.

14. A method of predicting branches, said method comprising:

storing a first set of partial prediction information within a first storage including a first plurality of locations;

storing a second set of partial prediction information within a second storage including a second plurality of locations;

performing a first hash function on input branch information to generate a first index for accessing a selected location within said first storage and performing a second hash function on said input branch information to generate a second index for accessing a selected location within said second storage, wherein said input branch information includes address information corresponding to a fetch address of a current branch instruction;

providing a prediction value based on corresponding partial prediction information in said selected locations of said first and said second storages; and

updating said selected locations of said first and said second storages dependent on whether said prediction value yields an accurate branch prediction.

15. The method as recited in claim 14, wherein said prediction value provides a strongly/weakly taken/not taken branch prediction indication that is indicative of whether the current branch instruction is taken upon execution.

16. The method as recited in claim 14, wherein said input information includes branch history information corresponding to an outcome of a number of preceding branch instructions.

17. The method as recited in claim 16 further comprising each of said first hash function and said second hash function operating on a portion of said branch history information.

19. The method as recited in claim 14 further comprising each of said first hash function and said second hash function operating on a different subset of bits of said branch address information.

20. The method as recited in claim 14, wherein each of said first and said second sets of partial prediction information includes a plurality of counter values each corresponding to a strongly/weakly taken/not taken branch prediction indication that is indicative of whether the current branch instruction is taken upon execution.

21.	The method as recited in claim 20 further comprising using said prediction value to determine whether the current branch instruction is taken upon execution and generating said prediction value by summing respective counter values stored within said selected location within said first storage and said selected location within said second storage

22.	The method as recited in claim 14 further comprising controlling whether a branch prediction is performed in accordance with a branch prediction hint encoded within the current branch instruction using said prediction value.

23.	The method as recited in claim 22, wherein each of said first and said second sets of partial prediction information includes a plurality of counter values each corresponding to a strongly/weakly agree/disagree indication that is indicative of whether said branch prediction hint bit embedded within said current branch instruction is to be used by said control unit.

24.	The method as recited in claim 23 further comprising generating said prediction value by summing respective counter values stored within said selected location within said first storage and said selected location within said second storage.

26.	The method as recited in claim 14 further comprising storing a third set of partial prediction information within a third storage including a third plurality of locations and performing a third hash function on said input branch information to generate a third index for accessing a selected location within said third storage

27.	A branch prediction mechanism comprising:

means for storing a first set of partial prediction information within a first storage including a first plurality of locations;

means for storing a second set of partial prediction information within a second storage including a second plurality of locations;

means for performing a first hash function on input branch information to generate a first index for accessing a selected location within said first storage and performing a second hash function on said input branch information to generate a second index for accessing a selected location within said second storage, wherein said input branch information includes address information corresponding to a fetch address of a current branch instruction; and

means for providing a prediction value based on corresponding partial prediction information in said selected locations of said first and said second storages; and

means for updating said selected locations of said first and said second storages dependent on whether said prediction value yields an accurate branch prediction.

IX. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

X. **RELATED PROCEEDINGS APPENDIX**

There are no related proceedings known to Appellant.